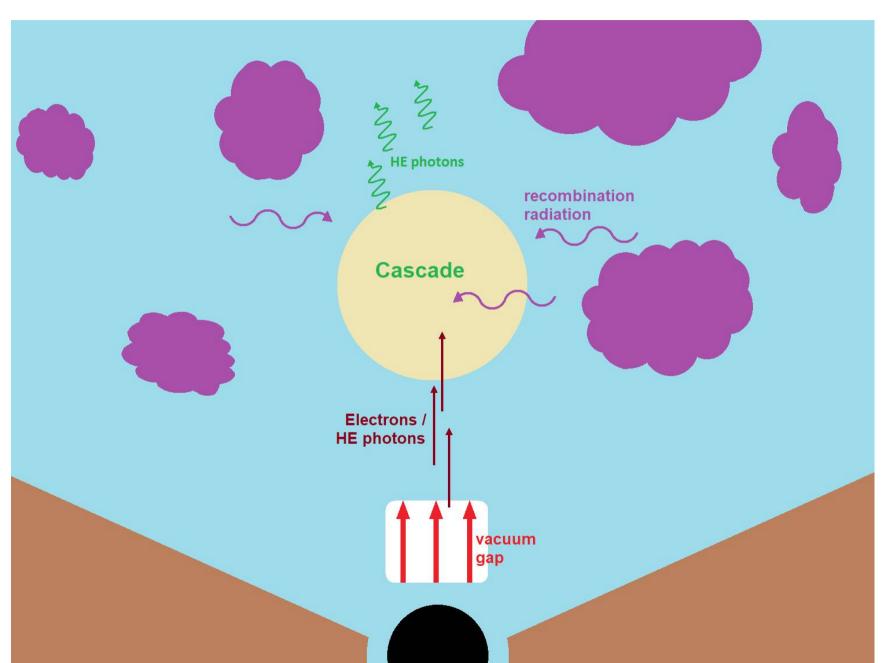
```
## Considering various inverse-Compton-scattering and pair-production formulae
import numpy as np
from scipy import integrate
from GeneralDefinitions import *
c = 290000000.0 # The velocity of light in units m/s
h = 6.6*10**(-34) # The Planck's Constant in units J*s
Pair cascades in active galactic nuclei
def nθPlanck(x,PlanckTemperature):# This gives the spectral number-density of the soft photons in units 1/m<sup>3</sup>.
    return (8*np.pi*me**3*c**3*x**2) / ( h**3 * (np.exp(x*me*c**2/(kB*PlanckTemperature)) - 1) )
x0Delta = 30*Theta
                                  # The situation of the Delta-peak. INPUT VALUE!
def n\Theta Delta(x):
                                  # This gives the number-density of the soft photons in units 1/m^3.
                                  # The coefficient for nO. INPUT VALUE!
    A = 10**(27)
    return A
                                  Talk by Christoph Wendel
if Usedn0 == n0Delta:
    x0 = x0Delta
                                  # This makes the following code easier.
    def COfAl(ga,gaP,n0):
                                  # Defining this COfAl separately is a trick to bypass the integration.
        return IntegrandOfAl(x0,ga,gaP,n0)
    In collaboration with Amit Shukla and Karl Mannheim
else:
    def C0fAl(ga,gaP,n0):
                                  # Equation Al.
        LowerIntBorder = max(x1,EAst0fA3(ga,gaP)/ga)
                                                        # Definition of lower integration border.
        if LowerIntBorder >= x0:
            LowerIntBorder=x0 # Prevent the lower integration border from exceeding the upper border.
        COfAlResult, COfAlError = integrate.quad(IntegrandOfAl, LowerIntBorder, x0, args=(ga,gaP,n0))
        return COfAlResult
ValuesForgammaP = np.logspace(np.log10(gaPminOf6(x0,gamma)),NumOfDecadesOfgamma,NumOfDecadesOfgamma*10+1)
ValuesForCOfAl = [COfAl(gamma i Usedno) for i in ValuesForgammaPLin] # Evaluate the integral. pl.figure(figsize=(12, 9), DPG spring meeting, Würzburg, 122.03.2018s final electron energy")
pl.plot(ValuesForgammaP, ValuesForCOfA1, label=LabelForIC)
                                                                      # Plot the function.
ax = pl.qca()
                                                                       # Get current axes.
ax xaxis set tick narams(lahelsize=16)
                                                                       # Set ticks
```





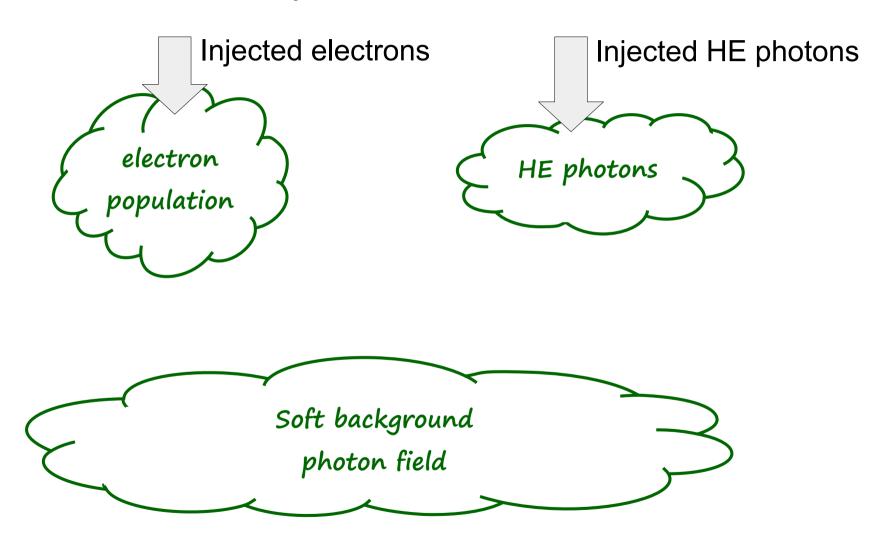
Cascades in AGN







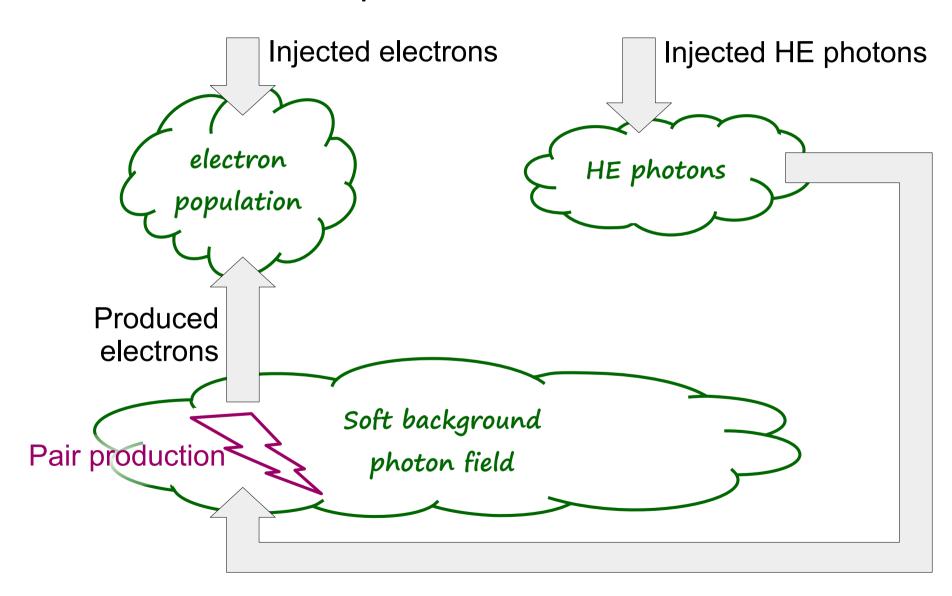
IC pair cascades







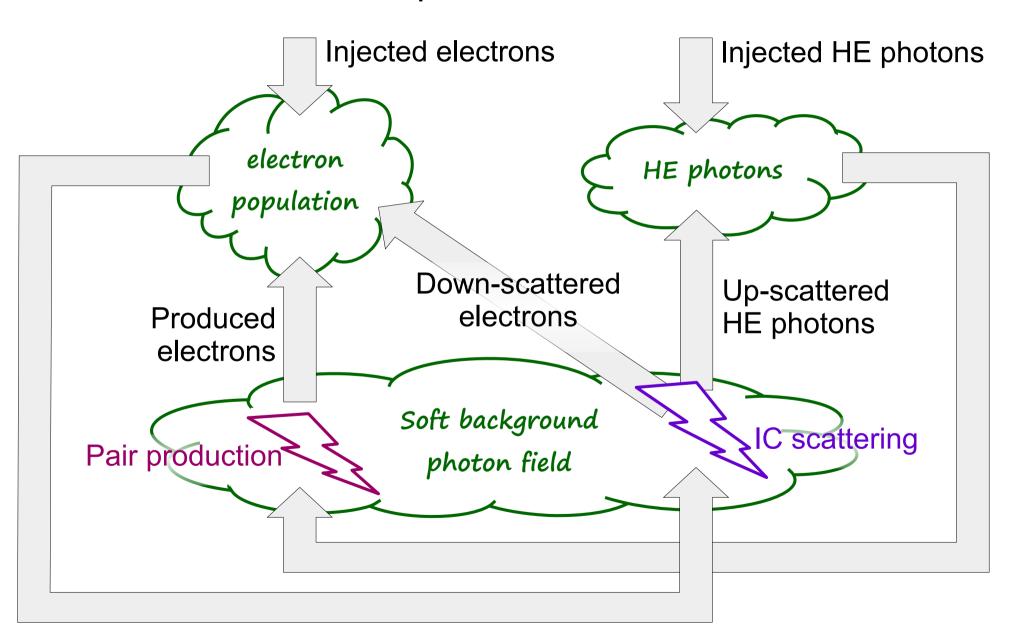
IC pair cascades







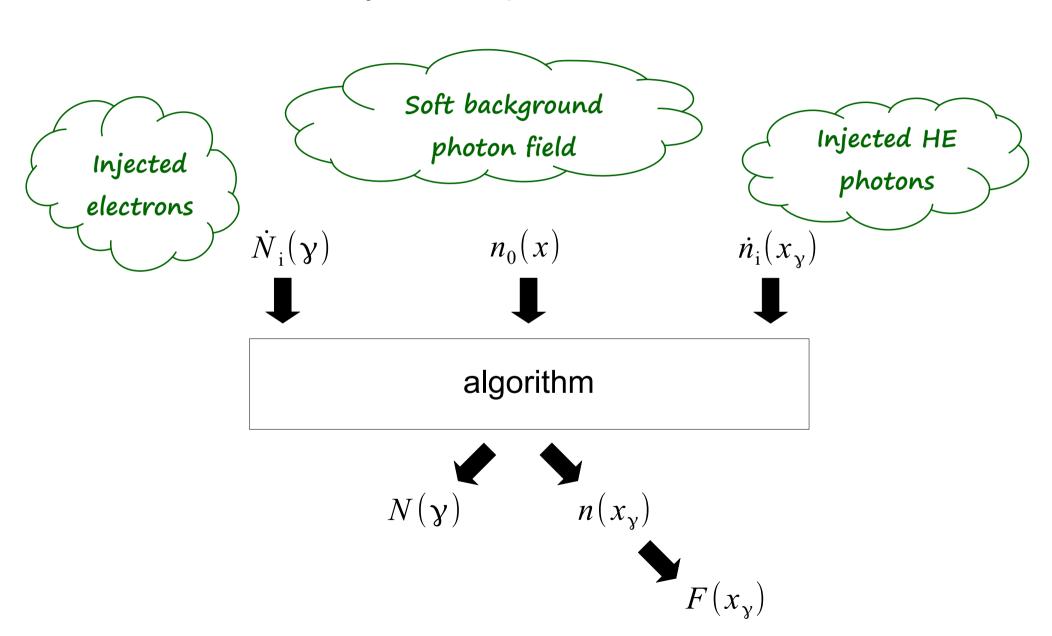
IC pair cascades







Python implementation



Lowestga and Hignestga in the ratio 9 to 1 in logarithmic space. NumberOfSamplingPointsForTotalRange = 100 # Number of sampling points. INPUT VALU

Number Of Sampling Points For Upper Range Python implementation oints For Upper Range

np.sort(np.append(np.logspace(np.log10(Lowestga),np.log10(Highestga),NumberOfSamplingPointsForTotalRange),np.logspace
DivisionPointgaLog,np.log10(Highestga),NumberOfSamplingPointsForUpperRange,endpoint=False))) # A sampling-range for

- This range is Specify input quantities: $\dot{N}_{\rm i}(\gamma)$, $\dot{n}_{\rm i}(x_{\gamma})$, $n_{\rm 0}(x)$
- # A sampling radio Determine $C(\gamma, \gamma')$ and $p(x_{\gamma}, \gamma)$ via $n_0(x)$
- NElectronsIterated = {} # A data-structure to save the findings of all the iteration steps. It will be a dictionary of distinguished
 - Determine electron distribution $N(\mathbf{y})$
- StartingTime tKN regime: Solve kinetic equation iteratively the point of time before the
 - while IterationCounter<21: $N_{ij}(\gamma) = \mathcal{F}(n_0, \dot{N}_i, \dot{n}_i, N_{j-1}, \gamma)$
- Values Forgal LogComput Thomson regime: Integrate continuity equation NElectrons Current NElectrons Dotni = UsedDotni and Data Dotni = UsedDotni = Use
- # Now, the interpolation of the computed values of the interpolation of the computed values of the interpolation values of the interpolation values of the interpolation value of the interpolating value of the interpolation value of the interpolating value of the interpolation value of the interpolation value of the interpolation value of the interpolation values of the interpolation
- Nelectrons Iterated ['ss. Iteration' 's Iteration Counter] = {}
 Interpolate Object Determine HE photon spectrum $n(x_{\gamma})$ [] =
 - # Now, evaluate the interpolated object and save the resulting np.array in the same dictionary
- Convert to flux density $F(x_{\gamma})$
 - # Now, the iteration:

CurrentNElectrons = InterpolatedObjectForValuesForNextNElectrons

IterationCounter +=1

Endinglime = time.time() # Now, determine the point of time after the end of the iteration.

TimeInterval = (EndingTime-StartingTime)/3600 # Determine the time in hours, that was spent by the

print("Time spent on the computation:". TimeInterval. "hours"



Specify input quantities



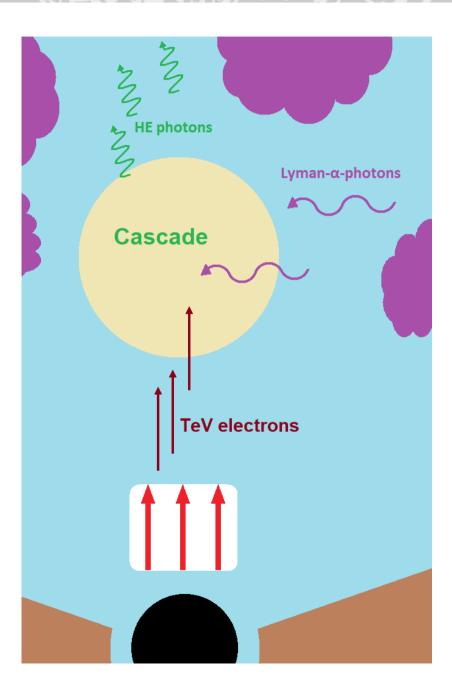
$$\dot{N}_{i}(\gamma) = \begin{cases} \text{Gaussian around } \gamma_{\text{mean}} & \text{if } \gamma_{1} \leq \gamma \leq \gamma_{0} \\ 0 & \text{otherwise} \end{cases}$$

$$\gamma_{\text{mean}} \approx \text{TeV}$$

$$n_0(x) = K_2 \cdot \delta_{\text{Dirac}}(x - x_0)$$

 $x_0 = h/(121.5 \text{ nm } m_e c)$

$$\dot{n}_{\rm i}(x_{\gamma}) = 0$$



Number Of Sampling Points For Upper Range Python implementation oints For Upper Range

Specify input quantities: $\dot{N}_{i}(\gamma)$, $\dot{n}_{i}(x_{\gamma})$, $n_{0}(x)$

A sampling-range for de logarithmic (γ, γ') the interval is used to the evaluation of the interpolate object of the evaluation of the interpolate of the polarithmic (γ, γ') and (x, γ, γ') the interval is used to the evaluation of the interpolate of the evaluation of the evaluation of the interpolate of the evaluation of t

Determine electron distribution $N(\gamma)$

StartingTime - tKN regime: Solve kinetic equation iteratively in the point of time before the

while IterationCounter<21: $N_{ij}(\gamma) = \mathcal{F}(n_0, \dot{N}_i, \dot{n}_i, N_{j-1}, \gamma)$ # Computation of NextNElectr $N_{ij}(\gamma) = \mathcal{F}(n_0, \dot{N}_i, \dot{n}_i, N_{j-1}, \gamma)$

Values Forgal Og Comput Thomson regime: Integrate continuity equation

InterpolatedObjectForValuesFo

• Determine HE photon spectrum $n(x_{\gamma})$

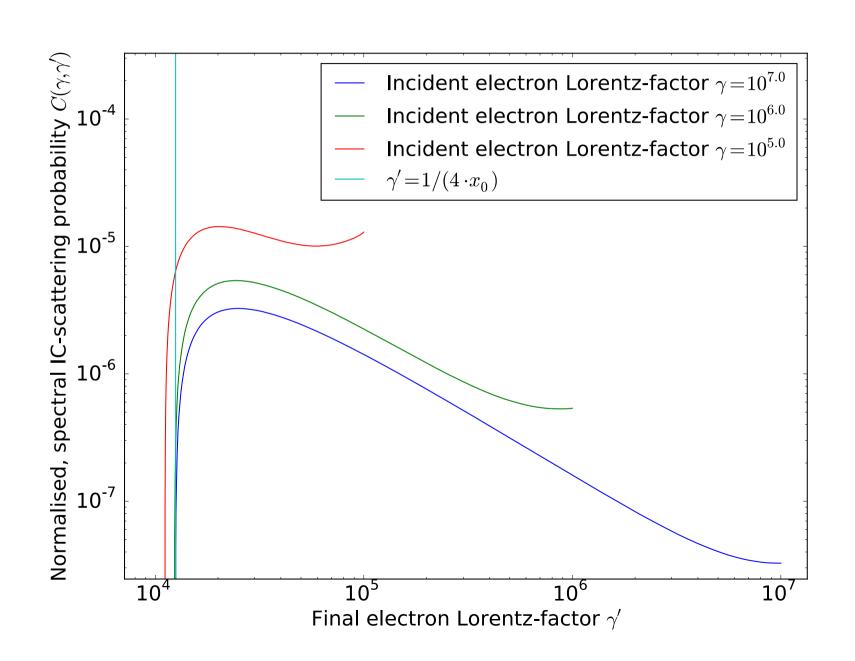
np.array in the same dictionary:

• Convert to flux density $F(x_y)$



Determine $C(\gamma, \gamma')$

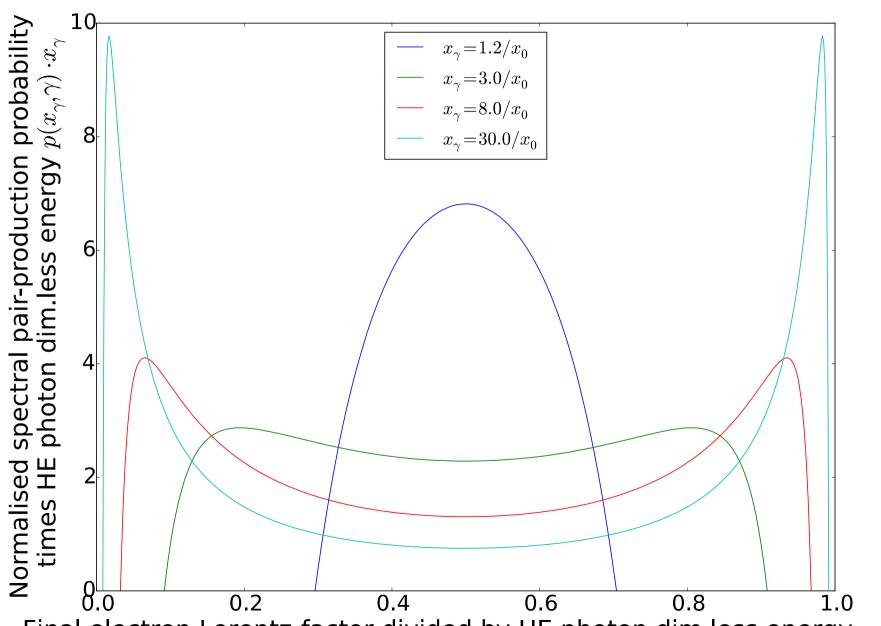






Determine $p(x_{\gamma}, \gamma)$





Final electron Lorentz-factor divided by HE photon dim.less energy γ/x_{γ}

Number Of Sampling Points For Upper Range Python implementation oints For Upper Range

np.sort(np.append(np.logspace(np.log10(Lowestga),np.log10(Highestga),NumberOfSamplingPointsForTotalRange),np.logspace
DivisionPointgaLog,np.log10(Highestga),NumberOfSamplingPointsForUpperRange,endpoint=False))) # A sampling-range for

Specify input quantities: $\dot{N}_{\rm i}(\gamma)$, $\dot{n}_{\rm i}(x_{\gamma})$, $n_{\rm 0}(x)$

A sampling-range $C(\gamma, \gamma')$ and $p(x_{\gamma}, \gamma)$ to $n_0(x)$ and $p(x_{\gamma}, \gamma)$ to $n_0(x)$ and $n_0(x)$

NElectronsIterated = {} # A data-structure to save the findings of all the iteration steps. It will be a dictionary of dictionaries

• Determine electron distribution $N(\gamma)$

def _EvaluateIteration()

StartingTime - tKN regime: Solve kinetic equation iteratively mine the point of time before the start of the iteration.

while IterationCounter<21: $N_{ij}(\gamma) = \mathcal{F}(n_0, \dot{N}_i, \dot{n}_i, N_{j-1}, \gamma)$ # Computation of NextNElectrons = np.array([NextNElectrons(i,Usedn0,CurrentNElectrons]])

Values Forgal LogComput Thomson regime: Integrate continuity equation NElectrons CurrentNElectrons

Now, the interpolation of the computed values of the interpolation values of the interpolation values of the interpolation value of the interpolating values of the interpolation value of the interpolation value of the interpolating values of the interpolation values of the in

Now, evaluate the interpolated object and save the resulting np.array in the same dictionary

• Convert to flux density $F(x_y)$

Now, the iteration:

CurrentNElectrons = InterpolatedObjectForValuesForNextNElectrons

IterationCounter +=1

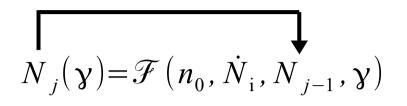
TimeInterval = (EndingTime-StartingTime)/3600 # Determine the time in hours, that was spent by the iteration the interval and the semperation in the iteration in the interval and the semperation in the iteration in the iteration in the semperation in the semperation in the iteration in the iter

print("Time spent on the computation:", TimeInterval, "hours

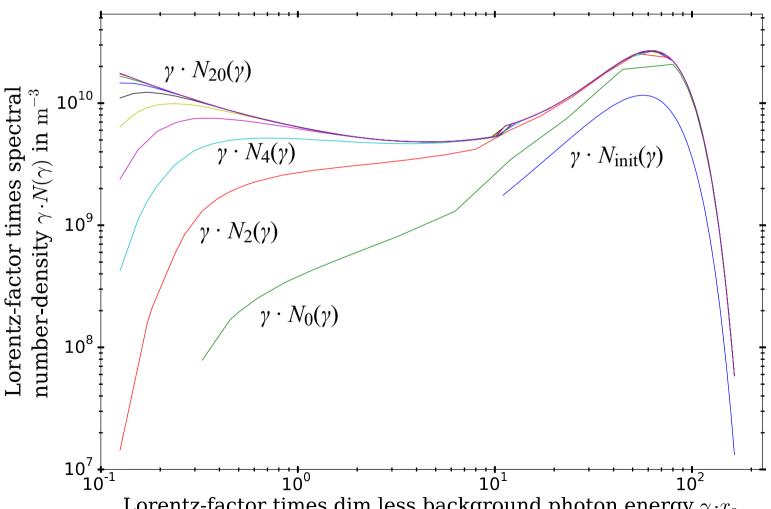


Solve cascade equation





 ${\mathscr F}$ from Zdziarski, 1988



Lorentz-factor times dim.less background photon energy $\gamma \cdot x_0$

Number Of Sampling Points For Upper Range Python implementation oints For Upper Range

Specify input quantities: $\dot{N}_{i}(\gamma)$, $\dot{n}_{i}(x_{\gamma})$, $n_{0}(x)$

A sampling-range $C(\gamma, \gamma')$ and $p(x_{\gamma}, \gamma)$ to $n_0(x)$ with $n_0(x)$ and $n_0(x)$

Determine electron distribution $N(\gamma)$

StartingTime - tKN regime: Solve kinetic equation iteratively in the point of time before the

while IterationCounter<21: $N_{ij}(\gamma) = \mathcal{F}(n_0, \dot{N}_i, \dot{n}_i, N_{j-1}, \gamma)$ # Computation of NextNElectr $N_{ij}(\gamma) = \mathcal{F}(n_0, \dot{N}_i, \dot{n}_i, N_{j-1}, \gamma)$

Values Forga Log Compute Thomson regime: Integrate continuity equation NElectrons = Current NElectrons, Dotni = Used Dotni and Data Thomson regime: Integrate continuity equation

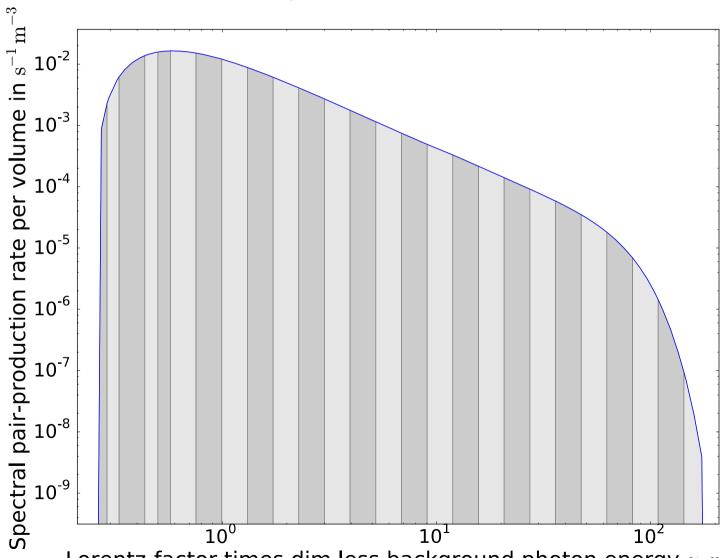
• Convert to flux density $F(x_{\nu})$



Integrate continuity equation



$$\frac{d(\dot{\mathbf{y}}(\mathbf{y})N(\mathbf{y}))}{d\mathbf{y}} = \dot{N}_{i}(\mathbf{y}) + \dot{N}_{PP}(\mathbf{y})$$



Lorentz-factor times dim.less background photon energy $\gamma \cdot x_0$

lowestga and Hignestga in the ratio 9 to 1 in logarithmic space. NumberOfSamplingPointsForTotalRange = 100 # Number of sampling points. INPUT VAL

Number Of Sampling Points For Upper Range Python implementation oints For Upper Range

np.sort(np.append(np.logspace(np.log10(Lowestga),np.log10(Highestga),NumberOfSamplingPointsForTotalRange),np.logspace
DivisionPointgaLog,np.log10(Highestga),NumberOfSamplingPointsForUpperRange,endpoint=False))) # A sampling-range for
ga. Logarithmic (Nision ff, the interval is usefited slightly below High estga) some addit onal points are inserted.

Specify input quantities: $\dot{N}_{\rm i}(\gamma)$, $\dot{n}_{\rm i}(x_{\gamma})$, $n_{\rm 0}(x)$

A sampling radio Determine $C(\gamma, \gamma')$ and $p(x_{\gamma}, \gamma)$ via $n_0(x)$

Determine electron distribution $N(\gamma)$

StartingTime - tKN regime: Solve kinetic equation iteratively the point of time before the

while IterationCounter<21: $N_{ij}(\gamma) = \mathcal{F}(n_0, \dot{N}_i, \dot{n}_i, N_{j-1}, \gamma)$ # Computation of NextNElectrons = np.array([NextNElectrons(i,Usedn0,CurrentNElectrons]])

Values Forgal LogComput Thomson regime: Integrate continuity equation NElectrons CurrentNElectrons

Now, the interpolation of the computed values of the interpolation of the computed values of the interpolation values of the interpolation values of the interpolation value of the interpolating value of the interpolation value of the interpolating value of the interpolation value of the interpolation value of the interpolation value of the interpolation values of the interpolation

NElectronsIterated['ss. Iteration' & IterationCounter] = {}
NElectronsIterated['ss. Iteration' & IterationCounter] = {}
InterpolatedObj Determine HE photon spectrum $n(x_{\gamma})^{\text{pect'}}$ =

Now, evaluate the interpolated object and save the resulting np.array in the same dictionary

• Convert to flux density $F(x_y)$

Now, the iteration:

CurrentNElectrons = InterpolatedObjectForValuesForNextNElectrons

IterationCounter +=1

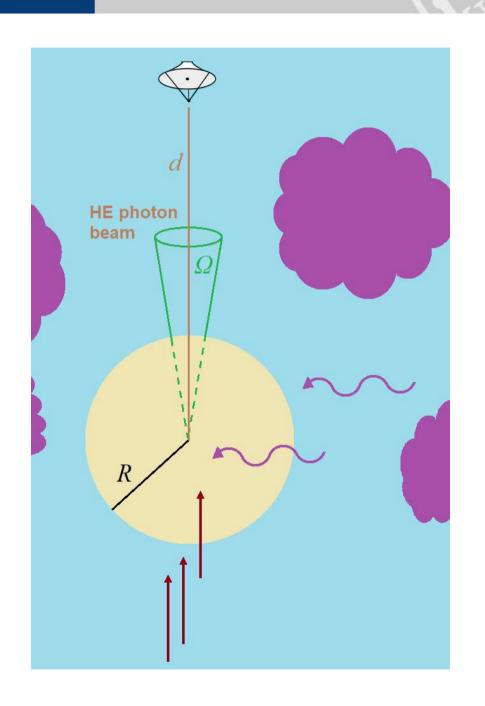
TimeInterval = (EndingTime-StartingTime)/3600 # Determine the time in hours, that was spent by the iteration are in the interval and the iteration in the itera

print("Time spent on the computation:", TimeInterval, "hours'



Determine HE photon spectrum





Spectral number density:

$$n(x_{\gamma}) = \mathcal{H}(n_0, \dot{n}_i, N, x_{\gamma})$$

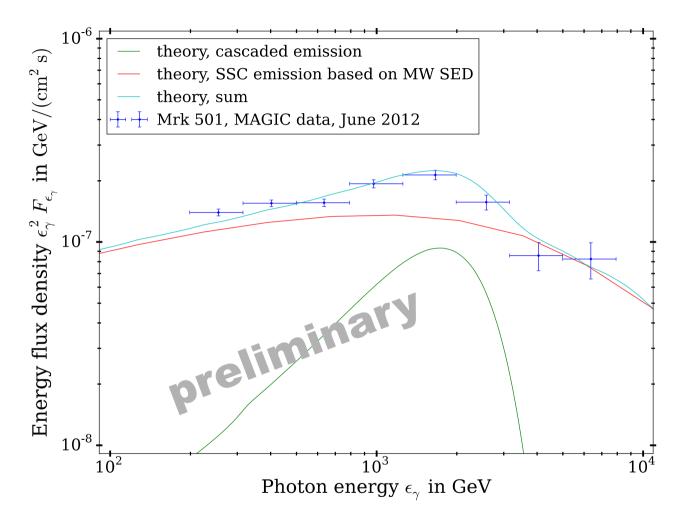
Convert to observed flux (vF_v):

$$F(x_{\gamma}) = \frac{4\pi c R^2}{\Omega d^2} n(x_{\gamma})$$



IC pair cascade in Mrk 501





Quantity	Used value
K_1	$1.80 \cdot 10^3 (\text{s} \cdot \text{m}^3)^{-1}$
K_2	$8 \cdot 10^{16} \mathrm{m}^{-3}$
$\gamma_{ m mean}$	$1.15 \cdot 10^{12} \text{eV/}(m_{\rm e} c^2 / e)$
σ	$0.80~\gamma_{ m mean}$
M	$5 \cdot 10^8 \mathrm{M_{\odot}}$
R	1.0 r _s
Ω	0.001 sterad

- SSC modelling by Amit Shukla with code by H. Krawczynski et al. 2004.
- Data processing by David Paneque et al. to be published

- Spectral signature of gap activity
- VHE generation partly within inner portion of AGN

Similar work:

- Jones, 1968, PhRv 167, 1159
- Zdziarski, 1988, ApJ 335, 786
- Mannheim & Biermann, 1989, A&A 221, 221
- Levinson & Rieger, 2011, ApJ 730, 123
- Wendel et al., 2017, AIPC 1792, 26